

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

December 20, 1994

3. REPORT TYPE AND DATES COVERED

Final Technical Report 8/1/93-9/30/94

4. TITLE AND SUBTITLE

Towards Increased Productivity of
Algorithm Implementation

5. FUNDING NUMBERS

AFOSR-91-0308

6/1/92 F
2304/FS

6. AUTHOR(S)

Robert Paige

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

New York University
Office of Sponsored Research
15 Washington Place
New York, NY 10003

8. PERFORMING ORGANIZATION
REPORT NUMBER

AFOSR-TR-95-0223

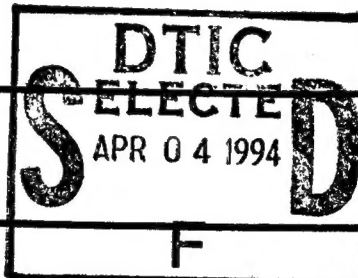
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Air Force /NM
Air Force Office of Scientific Research
Bolling AFB DC 20332

10. SPONSORING / MONITORING
AGENCY REPORT NUMBER

AFOSR-91-0308

11. SUPPLEMENTARY NOTES



12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

Unlimited

13. ABSTRACT (Maximum 200 words)

Last year we reported experimental results in productivity of nonnumerical algorithm implementation using transformational programming. Comparative benchmarks suggest that productivity can be increased by our approach over conventional hand crafted programming by factors ranging from 5.1 to 9.9. Preliminary results also showed that the running time of C code produced by this new approach can be as fast as 1.5 times that of tightly coded high quality Fortran.

19950403 026

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION
OF REPORT

Unclassified

18. SECURITY CLASSIFICATION
OF THIS PAGE

Unclassified

19. SECURITY CLASSIFICATION
OF ABSTRACT

Unclassified

20. LIMITATION OF ABSTRACT

SAR

Report Type: Annual Technical Report

Title: Towards Increased Productivity of Algorithm Implementation

GRANT: AFOSR-91-0308

August 1, 1993 to September 30, 1994

Robert Paige

Dept. of Computer Science
New York University/ Courant Institute
251 Mercer St.
New York, NY 10012

ABSTRACT:

Last year we reported experimental results in productivity of nonnumerical algorithm implementation using transformational programming. Comparative benchmarks suggest that productivity can be increased by our approach over conventional hand crafted programming by factors ranging from 5.1 to 9.9. Preliminary results also showed that the running time of C code produced by this new approach can be as fast as 1.5 times that of tightly coded high quality Fortran.

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

(a) Objectives

The proposed research aims to improve the production rate and the reliability of high performance implementations of nonnumerical algorithms. The approach utilizes program transformations that capture broad algorithm design principles. These transformations are evaluated prior to an implementation by testing whether they can be used effectively both to explain complex algorithms, and also to help design new algorithms. The implementation methodology makes use of conditional rewriting together with logic based program analysis. The program development methodology is evaluated by productivity experiments.

(b) Status

Last year was spent visiting DIKU at the University of Copenhagen. The Computer Science Department there is particularly strong in programming languages, especially in the subareas of type theory and partial evaluation. Interactions with Fritz Henglein and his student Jacob Rehof in type theory and with Neil Jones in partial evaluation provided valuable insight for scaling up the experiments in our AFOSR-funded research in productivity improvement of algorithm implementation.

The productivity experiments carried out in 1993 were reported in Dec. 1993 at the ACM SIGSOFT Conf. [Cai93a]. The results suggest that productivity of algorithm implementation in C (measured in terms of number of source C lines per unit of manual programming time) can be increased by our approach over conventional hand crafted programming by factors ranging from 5.1 to 9.9. Preliminary results also showed that the running time of C code produced by this new approach can be as fast as 1.5 times that of tightly coded high quality Fortran.

The preceding work aroused attention at a Dagstuhl workshop last year, and an invited talk was requested for the joint PLILP/ALP Conf. in Sept. 94 in which our program development technology could be publically demonstrated [Paige94d].

As encouraging as this sounds, there are two hurdles that still need to be overcome before this novel program development technology can be practical. First, the technology needs to scale up. The SIGSOFT paper reported experimental development of 9 C programs, the largest of which is around 10 pages long. A much more credible case would be made with programs 50 pages long. A related problem is the slow rate at which C programs are mechanically produced from hand-coded high level specifications. 3.3 lines of specification are turned into 24 lines of C every minute. A plan is outlined in [Paige94d] for increasing this production rate 6000 times.

Much of the research over the last year was in preparation for scaling up the program development technology. At the heart of our transformational methodology is a type/subtype system that facilitates perspicuous mathematical specification of efficient data structures and algorithms without extraneous implementation detail. More importantly, such specifications lead to a set-theoretic language with computational transparency; i.e., programs can be analyzed accurately for run-time resource utilization, so that programming can be guided by complexity considerations.

Most of our effort last year (as part of the scale-up effort) was in extending the type/subtype system with an abstract datatype capability and a crucial, but technically difficult, ability to handle recursively defined subtypes with disjoint alternation (which considerably broadens the

contexts in which a unit-time implementable associative access can be used in our specification language). This theoretical work is now completed, and the more pragmatic research involved in fitting it into a programming language has just begun. Since reading high level external input is the beginning of computation, we decided to begin this work with the development of algorithms to translate high level external input data in string form into the efficient data structures modeled by the new type/subtype system. This work is now done, and the core algorithms are reported in [Paige94b]. The next step is to develop new type and subtype inference algorithms for the specification language, and to modify our specification compilers in order to exploit the new type/subtype system and to generate efficient C code.

Sagiv, a postdoc on the AFOSR grant, spent most of the year working on efficient interprocedural analysis methods for distributive functions, and a paper coauthored with Tom Reps and Susan Horwitz has been accepted at POPL 95. That work will, hopefully, contribute to our plans to design interprocedural type and subtype inference algorithms. Jacob Rehoff, a student at DIKU supported by AFOSR, worked on the very interesting notion of optimal dynamic type checking and type conversion in a dynamically typed language. The completion of his work may shed light on the right kind of balance needed in a high level programming language between the flexibility of dynamic typing (as in Scheme, SETL, and in standard mathematical discourse) and the safety of strong typing (as in ML and in our current specification language).

Last year we were involved in several algorithm design projects with the aim of implementing these new algorithms as part of experiments with scaled up programs. In [Cai94] we extended Paige and Tarjan's simple algorithm [SICOMP 87] to find duplicate strings contained in a multiset of strings into a general purpose 'multiset discrimination' tool for finding duplicate data contained in a wide subset of our new type system. Using this tool, we were able to obtain improved solutions to virtually every compilation task from front-end macro processing down to global optimization by strength reduction. This tool was central to the development of the reading algorithms in [Paige94b].

With Nils Klarlund at the U. of Aarhus we developed new DFA minimization algorithms for large alphabet sizes and for potentially small number of transitions leading out from any state. These algorithms make use of new improved BDD simplification algorithms that depend on multiset discrimination. Klarlund needs these algorithms to speedup his program verification system.

With Neil Jones at the U. of Copenhagen we developed new unification algorithms that accelerate the classical union/find algorithm of Huet, and Vitter and Simons. Our algorithms are also top-down, and report occurs checks early. We plan to generate C code from high level specifications of these algorithms, and to conduct comparative benchmarks with implementations of other algorithms.

With J.P. Keller we derived a new DFA minimization algorithm with space asymptotically better than Hopcroft's algorithm of 1971. This research also involved integration of program transformations using APTS with set theoretic proof construction and checking using Keller's Aetna system. One conclusion drawn from this work is that the more rapidly that APTS can produce high performance C code, the more critical it needs to be equipped with the safety of machine-checked verification of the transformations involved in code production. Because we observed how difficult and painfully slow it is to construct such formal proofs (a view shared by

expert users of other systems), the formidable problem of making proof construction easier is essential to making our technology scale up in a practical way.

Besides the algorithms just mentioned, the new algorithms that we derived in [Bloom94] are candidates for scaled up experiments.

Last year we reported several other peripheral aspects of the AFOSR-funded research. An independent application of APTS as part of a running geometric constraint solving system is in press [Bouma93]. A survey of our research together with relevant open problems was published in [Paige94c].

(c) Publications

[Bouma93] Bouma, W., Cai, J., Fudos, I., Hoffmann, C., and Paige, R., "A Geometric Constraint Solver," Purdue U. TR 93-54, accepted CAD, 1993.

[Bloom94] Bloom, B. and Paige, R., "Transformational Design and Implementation Of a New Efficient Solution to the Ready Simulation Problem," accepted at Science of Computer Programming, 1993.

[Cai94] Cai, J. and Paige, R., "Using Multiset Discrimination To Solve Language Processing Problems Without Hashing," accepted at Theoretical Computer Science; also DIKU-TR Num. 94/16.

[Cai93a] Cai, J. and Paige, R., "Towards Increased Productivity of Algorithm Implementation," Proc. ACM SIGSOFT, in Software Engineering Notes, ed. David Notkin, Vol 18, Num 5, Dec. 1993, pp. 71 - 78.

[Paige94b] Paige, R., "Efficient Translation of External Input in a Dynamically Typed Language," to appear Proc. IFIP Congress 94 - Vol 1, eds. B. Pehrson and I. Simon, Elsevier, Sept 1994, pp. 603 - 608.

[Paige94c] Paige, R., "Atlantique Research Overview," Proc. Atlantique Workshop on Semantics Based Program Manipulation," eds. N. Jones and C. Talcott, DIKU Report 94/12, Paige.

[Paige94d] Paige, R., "Viewing a program transformation system at work" Joint 6th Intl. Conf. on Programming Language Implementation and Logic Programming (PLILP) and 4th Intl. Conf. on Algebraic and Logic Programming (ALP), eds. M. Hermenegildo and J. Penjam, LNCS Num.844, Springer-Verlag, Sep. 1994, pp. 5 - 24.

(d) Personnel

Jacob Rehoff, grad. student
email: rehoff@diku.dk
Address:
DIKU
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen East
Denmark

Shmuel Sagiv, postdoc, Associate Research Scientist, Sept. 1993 to May, 1994.
email: sagiv@diku.dk
Current Address:
Computer Sciences Dept.
U. of Wisconsin
1210 W. Dayton
Madison, WI 53706

Robert A. Paige, PI.
email: paige@diku.dk
office phone: 212-998-3080
Address:
DIKU
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen East
Denmark

(e) Interactions

i. Invited Talks

Seminar, U. of Copenhagen, "Reading Revisited," Sept. 1993.

Seminar, U. of Utrecht, "Towards Increased Productivity of Algorithm Implementation," Oct. 1993.

Seminar, SUNY, Albany, "Towards Increased Productivity of Algorithm Implementation," Dec. 1993.

Seminar, ISI/USC, "High Level Reading and Data Structure Compilation," Dec. 1993.

Seminar, UCLA, "Program Development and Algorithm Design by Transformation," Dec. 1993.

Seminar, Stanford U., "High Level Reading and Data Structure Compilation," Dec. 1993.

Seminar, Aarhus University, "Derivation of Fast Strong Bisimulation and DFA Minimization," Mar. 1994.

Theory Seminar, Aarhus University, "From Regular Expressions to DFA's," Mar. 1994.

Seminar, LITP, "Algorithm Design and Implementation by Program Transformation," Mar. 1994.

Seminar, Ecole Polytechnique, "Algorithm Design and Implementation by Program Transformation," Mar. 1994.

Seminar, U. of Trondheim, "Productivity Improvement of Algorithm Implementation," Aug. 1994.

Seminar, Tech. U. of Berlin, "Algorithm Design and Program Development by Transformation," Aug. 1994.

Seminar, Tech. U. of Berlin, "High Level Reading," Aug. 1994.

Seminar, U. of Saarlandes, "High Level Reading," Aug. 1994.

ii. Workshop and Conference Presentations

Presentation at ACM SIGSOFT Conf., "Towards Increased Productivity of Algorithm Implementation," Dec. 1993.

Presentation at IFIPS WG2.1 Meeting, "Algorithm Invention By Transformation," Rencum, The Netherlands, Jan. 1994.

Presentation at IFIPS WG2.1 Meeting, "High Level Reading and Data Structure Compilation," Rencum, The Netherlands, Jan. 1994.

Presentation at IFIPS WG2.1 Meeting, "Algorithm Specification," Rencum, The Netherlands, Jan. 1994.

Presentation at Dagstuhl Seminar on Incremental Computation and Dynamic Algorithms, "Algorithm Derivation and Program Development by Transformation," Schloss Dagstuhl, Ger., May, 1994.

Presentation at Dagstuhl Seminar on Incremental Computation and Dynamic Algorithms, "Towards Increased Productivity

of Algorithm Implementation" and an APTS System Demonstration,
Schloss Dagstuhl, Ger., May, 1994.

Presentation at Semantique Workshop on Semantics Based
Program Manipulation, "High Level Reading," Aarhus, Denmark, July, 1994.

Presentation at IFIPS Congress, "Efficient Translation of
External Input in a Dynamically Typed Language," Hamburg, Ger.,
Aug. 1994.

Presentation at AFOSR Software and Systems Program Workshop,
"Towards Increased Productivity of Algorithm Implementation,"
Washington D.C., Sep. 1994.

Invited speaker for Joint 6th Intl. Conf. on Programming
Language Implementation and Logic Programming (PLILP) and 4th
Intl. Conf. on Algebraic and Logic Programming (ALP), "Viewing
A Program Transformation System at Work," Sept., 1994, Madrid, Spain.

iii. Scientific Interactions with other Laboratories and DOD Projects

As an American member of Atlantique, a joint NSF/ESPRIT funded project to promote collaboration between American and European researchers in "Semantics-Based Program Manipulation," I spent last year at DIKU, the U. of Copenhagen. While there I interacted with Neil Jones, a pioneer in partial evaluation methodology, and other programming language researchers in his group. As a result, Neil and I did joint work on the acceleration of the classical union/find method with an application to unification. This work will be included in scaled up productivity experiments. Neil Jones is also guiding a graduate student in Copenhagen in a dissertation on the partial evaluation of SETL2. I am working with a graduate student at NYU on a specific form of SETL2 partial evaluation with the aim of speeding up applications in APTS (which would improve the turnaround time on the productivity experiments).

I also collaborated with Nils Klarlund at the U. of Aarhus on some of his problems in program and hardware verification using BDD technology. We produced a new theoretically faster DFA minimization algorithm for state-sparse but arbitrarily large alphabets using BDD's. The new algorithm will also be part of our scaled-up experiments.